EXHIBIT 29

CISCO SYSTEMS

Document Number	EDCS-517457, PRT # 23066
Based on Template	EDCS-189230 Rev 16
Created By	Sobha Kondaveeti
Author	Prasad Nune, Shilpi Saran, Sobha Kondaveeti, Atul Kumar

GSR VPLS Phase3
Software Functional Specification

Reviewers

Department	Name/Title
Development Engineering	Ravi Amanaganti, Padmaja Penumarthy, Russell Gardo, Sam Henderson, Vinit Bansal, Rama Paduvalli, Vijay Kulkarni
DevTest Engineering	Sriram Chandrasekaran, Manoj Devnani, Jyoti Khera, Krishna Eranti
Marketing	Rama Sekhar

Modification History

Revision	Date	Modifier	Comments
~~1~~	~~05/05/2006~~	~~Sobha Kondaveeti~~	~~Initial version~~
2	05/09/2006	Atul Kumar, Prasad ~~Nune, Shilpi Saran,~~ Sobha Kondaveeti	Updated sections based on internal review on 05/08/06
~~3~~	~~05/22/2006~~	~~VPLS SW team~~	~~Updated sections based on external review on 05/11/06, PRT # 23066~~
~~4~~	~~06/26/2006~~	~~Sobha Kondaveeti~~	~~Updated sections based on external review comments~~
5	06/27/2006	Sobha Kondaveeti	Modifying document approvers list
~~6~~	~~01/15/2007~~	~~Atul Kumar~~	~~Add information for L2 ACL and update bridge-domain section~~

Table of Contents

1 Problem Definition

GSR VPLS phase 3 development is targeted to enrich the VPLS solution on GSR by adding new features that will accelerate VPLS solution deployment. The release vehicle for these enhancements is IOS 12.0(33)S and it will encompass the following new features on top of existing VPLS feature-set:

- Hierarchical–VPLS (hence after, referred to as H-VPLS) capability using Access Pseudo-Wire.
- Quality of Service enhancements: The following new QoS features will be added:
  - Access PseudoWire QoS
  - 'match vlan' functionality for VFI AC interfaces
- MAC table management enhancements
- Layer 2 Access Control List support on Engine 5 for fugu-based Ethernet SPAs.

The below 'Features/usability matrix' table shows 'what feature lies where'. VPLS phase 3 enhancements target H-VPLS solution addition as well as support few new features that will make GSR VPLS solution more competitive.

Table 1: Features/Usability Matrix

	Needed to support H-VPLS	Enhancements for VPLS in general	Orthogonal to VPLS functionality
H-VPLS forwarding	☑		
Access-PW QoS	☑		
"match-vlan" for VFI AC intfs		☑	
MAC table enhancements		☑	
E5 L2 ACL			☑

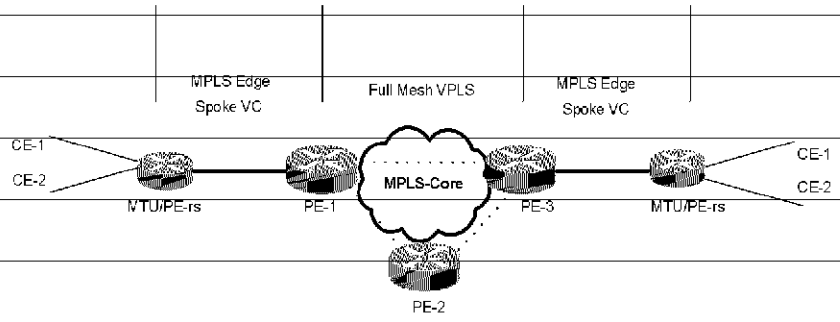
2 H-VPLS Overview

Virtual Private LAN Services (VPLS) (See EDCS-261721 for functional details for GSR VPLS) is an extension to the MARTINI-ENCAP draft and provides a mechanism for transporting Ethernet/802.3, VLAN [802.1Q] and VLAN-in-VLAN [Q-in-Q] traffic across multiple sites that belong to the same L2 broadcast domain. VPLS is a simple way to offer Virtual LAN services, including the appropriate flooding of Broadcast, Multicast and unknown unicast destination traffic over MPLS pseudo wires (LSPs), without the need for address resolution servers or other external servers.

Existing VPLS solution requires a full mesh of tunnel LSPs between all the PE routers that participate in the VPLS service. For each VPLS service, n*(n-1)/2 PWs must be setup between the PE routers. While this creates signaling overhead, the real detriment to large scale deployment is the packet replication requirements for each provisioned VCs on a PE router.

H-VPLS is a network topology proposal to reduce the number of pseudo wires within the MPLS network. H-VPLS reduces signaling and replication overhead to allow large scale deployment. The VPLS core PWs (Hub) are augmented with access PWs (Spoke) to form a two tier Hierarchical VPLS (H-VPLS). PW access to VPLS bridge domain is a predominant way of achieving hierarchical VPLS. One of the main features this project to support H-VPLS using non ethernet Access Networks. Please refer to Section 10 in http://www.ietf.org/internet-drafts/draft-ietf-l2vpn-vpls-ldp-07.txt for a complete discussion on H-VPLS and concepts of using access PW for doing H-VPLS.

Fig. 1: H-VPLS topology



3 Features Description

3.1 H-VPLS using Access PW

As noted earlier in the overview section, one of the preliminary ways to achieve H-VPLS is to use PseudoWire as an access type. It entails that layer 2 frames that need to be carried over the service providers' MPLS cloud transparently can arrive on a MPLS-enabled interface which starts to act as an attachment circuit interface. The decision as to which MPLS-enabled interface becomes the PW-access interface will be made automatically by the 'neighbor' configuration command options in the 'l2 vfi' configuration (explained in more details in the CLI section). Please note that with access PW implementation, a VFI domain can have a combination of Ethernet attachments circuits (port/dot1q/QinQ) and a PseudoWire attachment circuit.

3.1.1 Linecard generation limitations in a H-VPLS box

Access PW will be supported on GSR Engine5-based line cards and ***partially on Engine3 Gigabit Ethernet (Tetra) line cards***. Both Engine5 and Engine3 line cards will be supported as ingress of access PW. But only Engine5 linecards will be supported as egress line card for H-VPLS functionality. Because of multicast MGID limitations, Engine3 will not be supported as an egress card in H-VPLS solution. Software will always force access PW establishment through Engine5 line card for access PWs. Access PW will not come up if no LDP path exists through Engine5 cards.

To reiterate the engine 3 linecards limited role for access PW:

- Engine 3 linecards (tetra) interfaces can receive tagged traffic (i.e. ingress access-PW traffic). This traffic will undergo then destination MAC lookup (i.e. the usual tasks for VPLS forwarding) BUT
- Engine 3 linecards interfaces can ***NOT*** act as the egress point of this kind of traffic

The above restriction translates to forcing the core-PW establishment through an E5 linecard ONLY for PW-access traffic. Needless to say, the behavior carries over to the case when traffic egresses out of a tunnel (the underlying interface for the tunnel needs to be an E5 interface and *not* an E3 interface).

Also, Engine3 cannot be a back up FRR path. This is an existing limitation for Core PWs and will still be the limitation for spoke PWs as well

3.2 QoS Enhancements

3.2.1 Access PW QoS

With access PW, the MPLS-enabled interface (that now acts as an attachment-circuit interface) can receive traffic corresponding to many access PWs (i.e. traffic with different VC labels arrive

at the same (sub)interface). A mechanism is desired to identify traffic corresponding to a particular VC label and apply QoS policies on that stream. Currently, the only 'match' criterion available for ingress MPLS traffic classification is the 'match' based on MPLS EXP bits. Clearly, this is not good enough to isolate different PW access traffic.

The access-PW QoS enhancements in 33S will introduce a new 'match' criterion under class-map to classify access pseudo-wire traffic. This will enable QoS per access-VC. This new match criterion is not supported on egress interface.

An ideal approach to implement per PW QOS would be to attach a QoS policy to an individual PW rather than to an interface (that will receive traffic corresponding to multiple PWs). This would require non-trivial infrastructure changes in the pseudowire template and QoS. Given the timeline of 12.0(33)S release, it has been ruled out to make those changes in the existing 12.0 infra. (apparently, development to this effect is underway in12.2S train).

To fit in the existing model of attaching a QoS policy to an interface (in this context, interface refers to all kinds of interface: untagged [main] and tagged [802.1Q and 802.1 Q-in-Q]), hierarchical policy and class-maps with new match criterion will be applied to the interface. To identify per PW traffic, the new 'match' criterion would be based on ***{neighbor-id, vc-id} tuple*** as it uniquely identifies one access-PW.

A sample configuration would look like something similar to what is in example below (caveat: this is not finalized yet and is subject to MQC team and parser police approval):

a) **Parent Class Map Definition**:
This would use the new match-criterion to isolate per VC label traffic:

```
class-map hvpls-pw-1
   match pseudowire 1.1.1.1 vcid 101    ←New 'match' criterion
class-map hvpls-pw-2
   match pseudowire 1.1.1.1 vcid 102
.
.
class-map hvpls-pw-256
   match pseudowire 100.1.2.3 vcid 5
```

b) **Child Class Map Definition:**
This would use the existing VPLS match-criteria to isolate the kind of traffic (e.g. known/unknown etc):

```
class-map unicast
   match destination-address mac vpls-known
class-map multicast
    match destination-address mac vpls-unkown
```

c) **Child Policy Definition:**

This would associate actions with the filter-class identified by the child class-maps. Eg:

```
policy police-pw-1
    class unicast
        police <rate1>
    class class-def
        police <rate2>

policy police-pw-2
    class multicast
        set exp <val1>
        set qos-group <val2>
    class class-def
        police <rate3>
.....
.....
policy police-pw-256
    class unicast
        police <rate>
    class class-def
        police <>
```

Please note that the possible actions associated with these class-maps primarily limited to only these actions:

- 2-rate color-blind policing
- Set MPLS exp
- Set QoS group id

d) **Parent Policy Definition:**

This would serve as the hierarchy policy definition tying-in the child policy with parent class-map definition:

```
policy hvpls-access-pw
    class hvpls-pw-1          ← identifies a unique access-PW traffic
        service-policy police-pw-1    ← identifies different traffic flows and actions
    class hpvls-pw-2
        service-policy police-pw-2
     .....
     .....
    class hpvls-pw-256
        service-policy police-pw-256
    class class-default
        <>
```


e) **Associating with an interface:**

The parent policy formed above would then be finally applied to an interface. This interface **has to be an ingress access-PW interface** for the policy to make sense.

int *gig1/1/0.1*
service-policy input *hvpls-access-pw*
….
….
int gig 1/1/1.1
service-policy input *hvpls-access-pw*

3.2.1.1 Constraints and Scope of implementation of PW-QoS

The following constraints need to be imposed on the class-map, policy definition in this context:

- Hierarchical parent class can have **only** [vcid, neighbor-id] match, no other match types allowed. Also, match on [neighbor, vcid] is allowed only with parent class in hierarchical policy. Child class cannot have match [neighbor, vcid].
- Hierarchical child class can have only match EXP, match vpls-known and match vpls-unknown as the match criterion.
- In the parent policy, parent classes should have no actions other than service policy
- This new match criterion [vcid, neighbor-id] would be supported only on input policy
- The possible actions associated with the policy are primarily limited to policing, setting mpls EXP and qos-group
- The pseudowire policy is only allowed on Engine5 MPLS tag enabled interfaces
- Pseudowire policies works for only Engine5 spoke pseudowire traffic

3.2.1.2 Scalability limitations

There are two system-wide limitations for GSR QoS in 12.0S:

- 256 class maps under policy
- 1024 class maps in a router

Effectively, the total QOS supported access PWs are limited to 256 per router if user configuration hits the first limit. Otherwise then 1K class maps would be the next limitation.

First limitation is currently the soft limitation only. We will try to increase the number of class maps per policy in testing and assess the system performance. But it's a best effort.

3.2.2 Match VLAN with VPLS mac classification

Match VLAN or subinterface grouping allows the user to group together multiple vlans (either a range or individual different vlans) so that they can share the same policy. The match vlan support for non vpls matches already exists in 32sy itself. Support for 'match vpls known, unknown, multicast' with 'match-vlan' feature will be added in 33s release for Engine5 attachment circuit interfaces. Here's an example of a policy map for a sub interface group.

a) **Class Map Definition:**

This would use the existing VPLS match-criteria to isolate the kind of traffic (e.g. known/unknown etc):

```
class-map unicast
   match destination-address mac vpls-known
class-map multicast
   match  destination-address mac vpls-unkown
```

b) **'match-vlan' Class Map Definition:**

```
class-map customer1
   match  vlan 1 2 3    ← where 1,2,3 belong to VPLS bridge domain.
                          All vlan-ids included should be configured with bridge-domain.
class-map customer2
   match  vlan  4 10-15 20
```

c) **Child policy Definition:**

Associate policy with the traffic type:

```
policy customer1  child  policy
   class unicast
      police <>
      bandwidth <>
   class multicast
      police <>
      bandwidth <>
   class class-def
      <>

policy customer2  child  policy
   class unicast
      police<>
      bandwidth <>
   class multicast
      police  <>
   class class-def
      <>
```

d) **Parent policy Definition:**


```
policy vpls-policy
  class customer1
    service-policy customer1_child_policy
  class customer2
    service-policy customer2_child_policy
.....
```

c) **Associating with an interface:**

This hierarchical policy containing the "match-vlan" can ***ONLY*** be applied to a main interface. It will match on those vlan-ids off that main interface (since vlan-ids are unique for a main interface only).

```
interface gig x/y/z      ← applied on a main interface
  service-policy input vpls-policy
```

3.3 MAC table management enhancements

Following MAC table enhancements will be done in this phase.

- Ability to add static MAC address to DMATM table
- Provide global configuration command to set per AC MAC limit (VFI limit is already available)
- A configuration option to shut VFI/AC when MAC table limit reaches

3.3.1 Static mac address addition

As per the EFT feedback from VPLS phase1 and 2 (documented in ddts# CSCsc22123), GSR needs a way to add statically a mac address in the mac address table (DMATM) for vpls. This will be done via a configuration command. Following are some of the main properties w.r.t these static mac addresses/CLI.

- The command should be nvgened.
- Static mac addresses should not age out.
- *"clear mac address-table ..."* should not clear these static entries.
- *"show mac address-table ..."* should show these address as static.
- Limit this command to Ethernet Access Circuit interfaces only (and not access-PW) for this release.

The syntax for this command is detailed in the CLI section later in the document.

3.3.2 Limit on number of MAC addresses per AC

As per the EFT feedback from VPLS phase1 and 2 (documented in ddts# CSCsb53185), GSR needs a way to limit the number of MAC addresses per attachment circuit. As in 12.0(32)S release, the ability to limit the number of MAC addresses per VFI is already available. This new command enhances the capability by having a

- command to set global AC limit. With this, all the ACs will get this configured limit.
- command to set per VFI AC limit. All ACs that belong to that VFI will get this limit.

If both global and VFI AC limits are set, then the VFI AC limit should take precedence. These commands should be nvgened.

3.3.3 Shut VFI/AC with MAC table limit

As per the EFT feedback from VPLS phase1 and 2 (documented in ddts# CSCsb53185), GSR needs a way to shut down the AC interface or the bridge-domain when the MAC address table limit for that AC/BD has reached. The existing command *"mac address-table limit ..."* will be enhanced to add another option to support this feature. Some things to consider for this operation:

- Should generate syslog for 'shut' operation.
- Should keep the VFI/interface in 'shut' state until the user manually does a "no shut".
- Show bridge domain should display this state.
- Provide this command only for bridge-domain and AC interfaces.

Please note that the requirements for this option are still under investigation.

3.4 Layer 2 ACL for E5

With the wide deployment of GSR Ethernet linecards in the metro Ethernet space, there are customer demands to have security tools at each level. Layer 2 access control lists provide a mechanism to have a layer of control and security at the frame entry-door level.

Layer 2 ACL allows to permit/deny a layer 2 ethernet frame based on the source MAC address on a per-interface level (an interface can be a main intf/dot1q/qinq). Note that even though this feature is targeted to go along with other VPLS enhancements, it is orthogonal to VPLS or L2VPN technology. However, it is expected that L2VPN deployed topology are the early adapters of this feature.

Salient points to consider for this feature:

- This feature can be applied to the granular level of main port, vlan, or a QinQ interface.
- The default catch all entry behavior of all L2-ACL is deny. This can be changed to permit by explicitly configuring "permit any" ACE at the end of an ACL.
- Deny all packets with broadcast or multicast MAC addresses as source MAC address.
- Both L2 and L3 ACL can be applied on the same interface.
- When port tunnel is enabled on the main interface then the L2ACL applied on the interface should be applicable to all the incoming traffic on that port.

In general, the feature support and functionality scope of L2 ACL implementation should match or exceed the existing implementation on E3 tetra linecards.

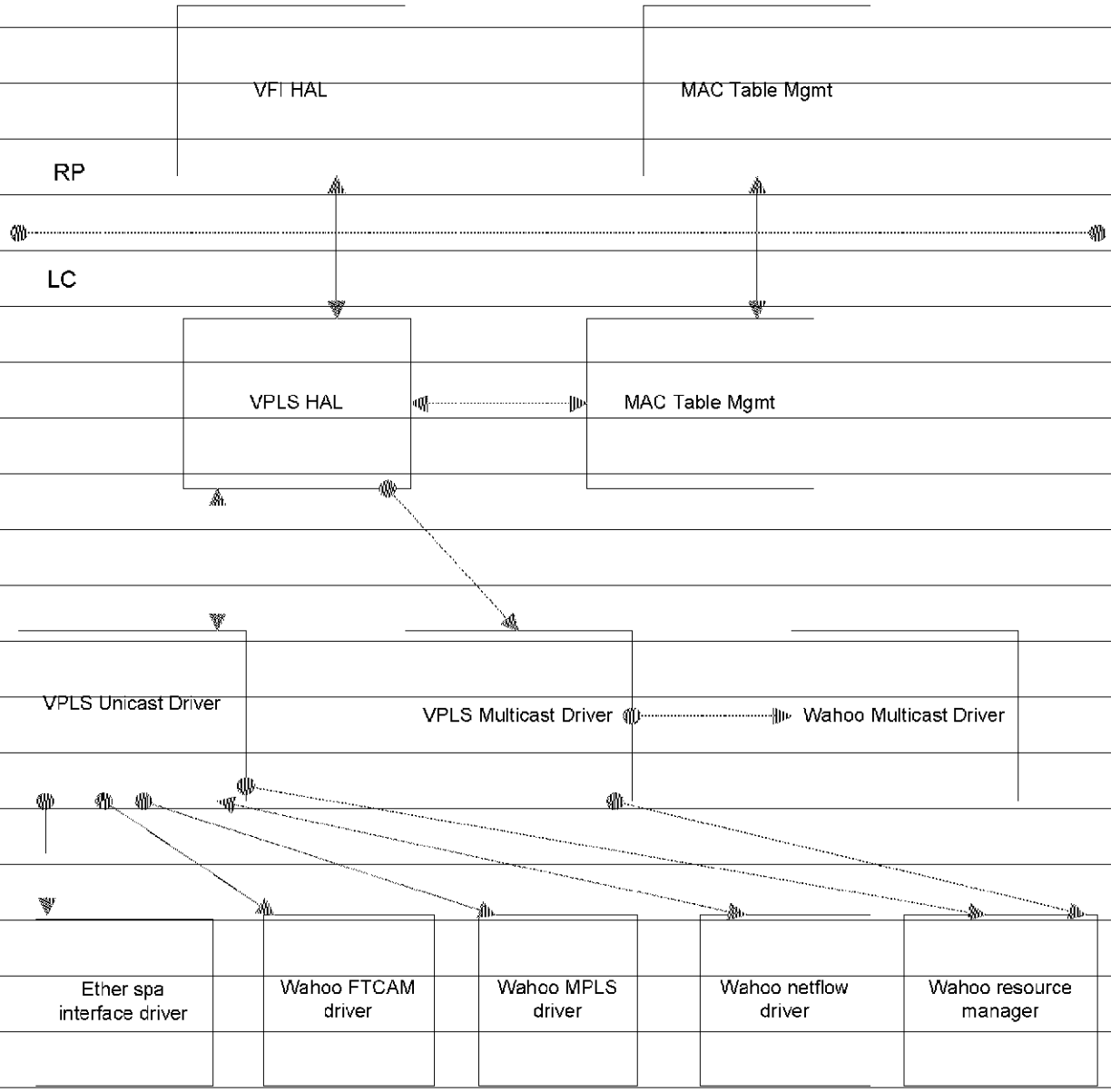
3.4.1 Limitations for L2 ACL

- This feature will be implemented in the layer 2 tcam present on fugu spa. It is **not** supported on gila-based versions of Ethernet SPAs.
- Support up to 5K order independent MAC ACEs per SPA (the limitation comes from the size of [port, SA] l2-tcam).
- Maximum number of distinct ACL lists that can be configured is 100. Ranges from 700-799.
- No MAC address mask support. So, it is basically a match on all the 48-bits and not section of bits as identified by a mask.
- Only Source MAC ACLs support. Destination mac-filtering is enabled by default in Ethernet broadcast medium for layer 3 packets.
- Only ingress ACLs support.
- No CAR or QOS support using these ACLs.

4 Software Architecture

Following figure shows a high level diagram of VPLS functional blocks. In the diagram, shaded modules will be extended to support access PW.

Figure 2 VPLS Modules



5 Software Requirements

5.1 H-VPLS using PW Access

The core PWs implements split horizon forwarding (i.e. packet received from MPLS network are not sent back into the MPLS network) towards the MPLS network to prevent loops. But access PW will operate in non split horizon mode since is used as an AC to the VFI. That is the packet received from access PW is sent to all ACs including other access PWs and core PWs except the

same access PW itself. So, from the user configuration perspective, the only difference between core PW and access PW is the split horizon mode.

5.1.1 CLI Changes

A ***'no-split-horizon'*** keyword with *neighbor* command will be enabled under VFI configuration to differentiate the access PWs. This will enable the PW to operate in non split horizon mode. A sample output of an access PW configuration will look like this.

l2 vfi VPLSA *manual*
vpn id 100
bridge-domain 10
neighbor 9.9.9.9 *encapsulation mpls*
neighbor 12.12.12.12 *encapsulation mpls*
neighbor 33.33.33.33 101 *encapsulation mpls no-split-horizon* ← Access PW1
neighbor 33.33.33.33 102 *encapsulation mpls no-split-horizon* ← Access PW2

5.1.2 MTU considerations

With the introduction of access PW to VFI there is a need to support "AC-less VFI", i.e. a VFI without Ethernet AC. In existing implementation, the MTU for bridge domain is derived from AC interface. In order to support MTU configuration for AC less VFI, a new 'mtu <value>' CLI will be introduced by PI code under 'l2 vfi' command chain.

An output of sample l2 vfi MTU configuration is given below.

Router(config)# l2 vfi <name> manual
Router(config-vfi)# mtu <1500-9180>

5.1.3 Shut/Un-shut bridge domain

A new place holder for per bridge-domain specific attributes is being introduced for GSR platform. There is also a request to provide the ability to shut or unshut a specific bridge domain either explicitly or as a result of MAC table limit breach. In 12.0(33)S, only [no] shut command is supported under this global bridge-domain command.

Router(config)# bridge-domain 10
Router(config-bd)# [no] shut

5.1.4 Control Plane

Existing ITD code in futurama branch supports 'no-split-horizon' key word. Following is the brief summary of changes in ITD code to support access PW.

- ITD platform code will provide an API to get MTU from vfi and provide notifications to MTU change. It will also query GSR bridge-domain specific code to see if a MTU change is allowed or not.
- An API to get the status of the bridge domain as well as the notification when it changes
- ITD code to bring up VFI without having an AC. Bring up the VFI only if bridge domain is configured under l2vfi. Bridge-domain has to be UP in that case.

5.1.5 GSR Platform Changes

- RP and LC code changes to force access PW through Engine5 line cards
- Extend VPLS HAL to process new AC bind/unbind requests of PW Access type
- Extend Engine5 Multicast driver to support access PW and FRR with access PWs.
- Extend Engine3/Engine5 unicast drivers to setup forwarding for Access PW
- Changes to Engine5 net flow driver code to have support for new learning profile and lookup.
- Changes to bridge domain code to have support for MTU configuration. In existing VPLS model, bridge domain configuration is allowed under an interface only. A new 'bridge-domain' command with shut/no shut options under main configuration will be introduced to configure MTU for bridge domain.

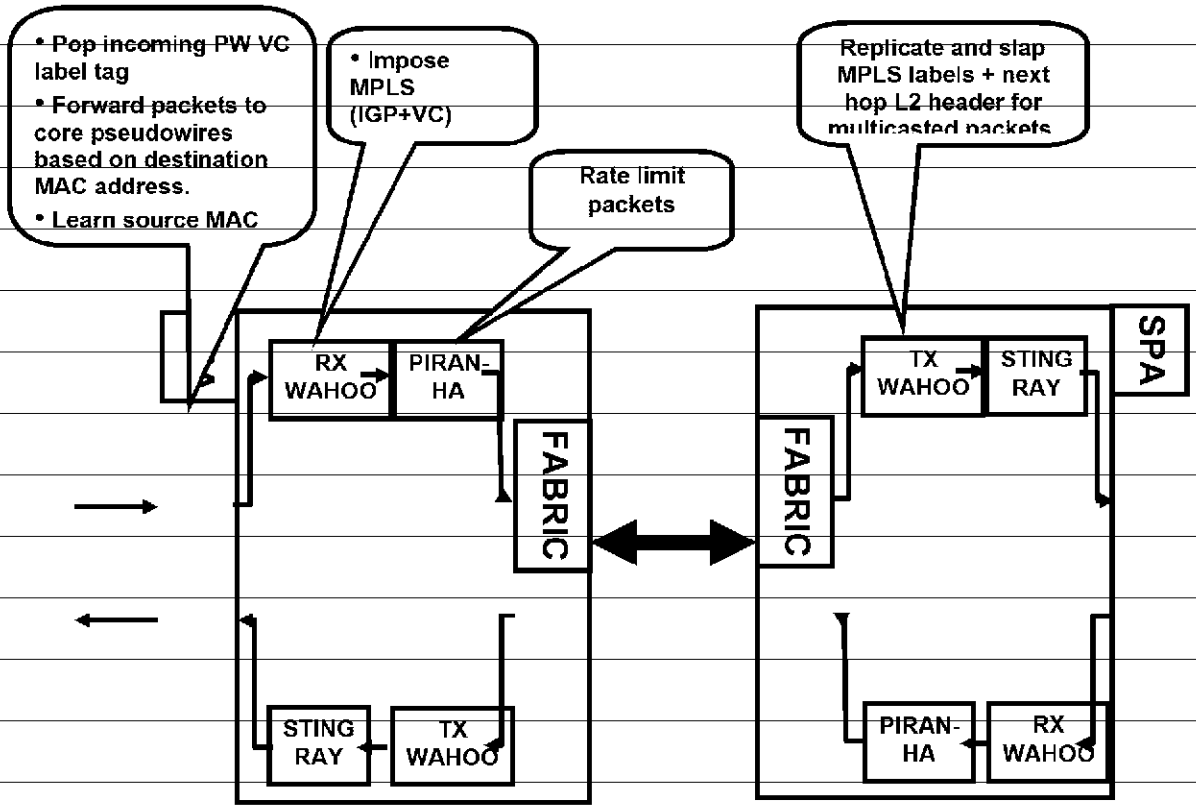
5.1.6 Data Plane

The packet arriving on access PW will be of the following format:

```
[4B VC label][Optional 4B control word][Ethernet L2 header]
```

The figure below shows the data plane forwarding architecture. What follows after is a brief description of the packet forwarding in H-VPLS paradigm. Please note that this is not a complete description of packet forwarding which is beyond the scope of this document, but this serves mainly to highlight the salient features and bring out the differences in H-VPLS forwarding.

Fig. 3: H-VPLS Forwarding Architecture



5.1.6.1 Unicast Forwarding and MAC Address Learning

Access PW unicast packet processing is very similar to the packet process done in current Rx Disposition path. The key difference would be the way the source MAC addresses are learnt for the packets received from Core PW and going out on Access PW. Note that in existing VPLS implementation, we always learn MAC addresses on edge facing cards (Rx Imposition, Tx Disposition). But with an access PW introduction, to minimize the code changes and simplify the packet processing for access PW, it is decided to do MAC learning on Rx cards (i.e. the card that first receives the packet in the box). If the packet is received from Access PW learning happens on Rx Imposition card and if the packet is received from core PW and going out on access PW, learning happens on Rx Disposition card. In both the cases, the rewrite is imposed on Rx side and Tx path takes regular MPLS code path. There will not be any changes to learning for existing AC and core PW packets.

It is possible to have packets arriving on access PW and going out on another access PW. This case is no different from regular unicast path.

There are no changes expected in Tx Imposition/Tx Disposition forwarding.

5.1.6.2 Multicast Forwarding

Current multicast implementation uses interface number check on Tx card to make sure that the packet does not arrive and go out on the same access circuit. But we cannot use interface number for access PWs since multiple VCs can go through the same interface. So we need to store incoming VC label in the replicord in order to achieve AC split horizon. But replicord does not have space to store VC label or some other form of id.

There are multiple options explored to implement this. Here is the brief summary of each one of them.

a) Increase the current replicord size from 48 bytes to 64 bytes. But this would be an approximate15% hit to an existing performance
b) Reduce the number of tags supported by one. But this would impair the FRR solution requires 4 tags
c) Existing multicast does not use PLU TCAM. Introduce a lookup using PLU TCAM to get VC label for split horizon check. This solution does not have any performance impact as such, but will involve significant driver and Prep ucode stage changes.

The final decision is to go with option (c) as this causes least impact on the existing functionality and performance.

In addition to the above mentioned change, an existing ucode in Prep/Mip/PoP stages will be extended to support access PW.

5.1.6.3 Explicit-null support

It is possible that the n-PE GSR device needs to advertise explicit null label also as part of access pseudowire setup. In that scenario, incoming packet on an access PW will have explicit-NULL on top of the VC label.

[4B Explicit-Null label] [4B VC label] [Optional 4B control word] [Ethernet L2 header]

GSR would need to pop off two labels in that case and do the mac forwarding based on the destination mac address further buried down in the frame.

5.1.7 Restriction

Following are few of the high-level restriction for H-VPLS software functionality:

- Multicast packet sub interface counters does not work for packets dropped on access PW because of design limitations
- Access PW establishment will be forced through Engine 5 linecards. There will be not be any configuration restriction for this but PW will not come up if the LSP path does not go through Engine5 card. Please refer to section 3.1.1 "Linecard Generation Limitations for a H-VPLS box'

6 Summary of features

Table 2: Summary of features

Feature	Engine5	Engine3	Comments
Access PW	YES	NO	Only ingress support on Engine3 cards.
Access PW FRR	YES	NO	Engine3 cannot be a backup FRR path for spoke PWs.
Link Bundling	NO	NO	
Access PW QOS	YES	NO	for unicast traffic only supported on ingress traffic only
Match VLAN	YES	NO	
Static MAC Address	YES	YES	Capability to add static MAC address to MAC table on a given bridge domain for given inter face. This is not supported for access PWs and core PWs as well.
AC MAC limit	YES	YES	Capability to limit number of MAC addresses learnt on AC. This is a global configuration for all ACs in the router.
shut option for VFI/AC when MAC table limit	YES	YES	Option to shut the VFI or AC when MAC ta ble limit reaches and informs the user through syslog. The VFI/AC remains in shut state until the user manually enables it.
PW redundancy	YES	YES	This feature is available in 32sy. test only effort. There are no software changes required.
L2 ACL	YES	YES	
Mechanism to disable	NO	NO	

Feature	Engine5	Engine3	Comments
forwarding between PW spokes			
H-VPLS with L2TPV3 edge	NO	NO	
nPE Core Optimization	NO	NO	
IGMP and PIM snooping support for VPLS	NO	NO	
Auto Discovery	NO	NO	

7 Scalability

The current scale target is tabulated below.

Table 3: Scalability Limits

Scalability	Engine5		Engine3	
	LC Limit	System Limit	LC Limit	System Limit
Access PW	2k	8K including both access and core PWs		
Core PW	5K	8K including both access and core PWs		
Access Circuits	2K	4K	1K	
Number of VFIs	2K	4K	1K	4K
Max MAC table limit	128K total (also per direction)	320K	64K (32K per direction)	320K
L2 ACL(1)	5K per SPA		5K	

(1) L2 ACL shares the MAC L2 TCAM with mac accounting feature. The total size of that tcam is 5120 entries. Hence the scalability number of L2 ACL feature also depends on the number of mac entries present for src mac accounting feature.

8 Memory and Performance Impact

Existing Engine5 multicast implementation itself is already beyond the ucode cycle limit to achieve line rate performance. Access PW implementation would further drops the multicast performance by an approximate 15%. More accurate numbers will be documented in the design document.

9 Packaging Considerations

No new hardware packaging consideration is involved in this project.

Software packaging consideration:

IOS 12.0(33)S image will be the release vehicle for these planned features. VPLS phase3 development will be based on futurama branch that contains the ITD changes related to access PW.

10 End User Interface

10.1 Access PW Configuration

A key word 'no-split-horizon' will be added to the 'neighbor' command in VFI configuration. This will enable the PW to operate in non split horizon mode.

1. [no] l2 vfi *name* manual
2. [no] vpn id *vpn id*
3. [no] neighbor *remote router id [vc-id]* {encapsulation {mpls} [no-split-horizon]
4. [no] shutdown

Table 4: Access PW configuration example

Command	Description
Router(config)#l2 vfi vpls1 manual	configures vfi
Router(config-vfi)#vpn id 10	assigns a vpn id
Router(config-vfi)#neighbor 4.4.4.4 100 encapsulation mpls no-split-horizon	access PW configuration with vcid 100
PE1(config-vfi)#bridge-domain 10	adding bridge domain to VFI

10.2 Shut configuration under bridge domain

A new CLI to configure MTU under bridge domain will be introduced to support "AC-less" bridge domain.

1. [no] bridge-domain *bridge-domain-id*
2. [no]shut

Table 5: Configuring shut for bridge domain

Command	Description
Router(config)#bridge-domain <id> Router(config)#[no] shut	configures shut/no shut for bridge domain

10.3 MAC Table options configuration

Table 6: Static MAC address configuration example

Command	Description
Router#(config) [no] mac adress-table static <H.H.H> interface GigabitEthernet <if num> bridge-domain <id>	Adds static MAC address into MAC table for a given bridge domain on given interface.

Table 7: AC MAC limit configuration

Command	Description
Router#(config) [no] mac adress-table limit bridge-domain <id> intf <name> maximum <number>	sets the limit on number of MAC addresses learnt on AC

Table 8: VFI shut configuration

Command	Description
Router#(config) [no] mac adress-table limit action <limit <flood |no-flood> | shutdown recover-interval #>	This sets global MAC table limit action and will be applied to all VFIs and ACs in the router. VFI will come to no shut state automatically after recover-interval time.
Router#(config) [no] mac adress-table limit <bridge-domain #> action <limit <flood|no-flood> shutdown recover-interval #>	This sets MAC table limit action for a given bridge domain or AC. VFI will come to no shut state automatically after recover-interval time.

10.4 Layer 2 ACLs

The CLI for Layer 2 ACL feature is given below. The CLI is same as for E3 linecards.

Table 9: L2 ACL configuration command

Command	Description
Router#(config) [no] access-list <700-799> {permit | deny} <address>	To create an access list based on MAC address.

Table 10: Command to apply L2 ACL to an interface

Command	Description
Router(config)# interface gig3/0/4.1	Enter the interface configuration mode
Router(config-subif)# [no] mac access-group <access-list number>	Apply MAC ACL to an interface

- Layer 2 ACL access list number is between 700-799
- All the ACEs are configured separately e.g.
  - access-list 701 deny 0.0.44
  - access-list 701 deny 0.0.45
  - access-list 701 permit 0.0.44 ← This becomes the backup ACL
  - access-list 701 permit 0.cccc.bbb5

- access-list 701 permit any

- Sample configuration example:

Router# config t

Router (config)# access-list permit 700 0003.fd1b.8700

Router (config)# access-list permit 700 0003.fd1b.8701

Router (config)# access-list permit 700 0003.fd1b.870a

Router (config)# access-list deny any <------- *This is optional. Default is deny.*

Router (config)# int gig 6/1/0.1

Router(config-subif)# mac access-group 700 in

Router(config-subif)# end

By default "deny any" entry will be added to the end of each ACL (all packets with no ACE entry in the ACL will be dropped). User can change this behavior by explicitly configuring the "any permit" entry in the acl.

10.4.1 Show command for layer 2 ACL

Table 11: Show command to display L2 ACL information

Command	Description
Router# show mac access-lists [<acl_num>]	Displays all the aces configured for all interfaces across every LC and the cumulative ace counters. When executed on the LC, it displays the information corresponding to that linecard only. When acl_number is specified, it displays the information for the ACEs of that acl number only.

11 Configuration and Restrictions

The configuration and restrictions applicable to each section is mentioned in the corresponding section itself. Because of the nature of contents, it helps in understanding if the constraints are listed along with the features other descriptions. It is deliberately avoided to lump them together and reproduce here to avoid information mismatch in the future.

12 Testing Considerations

Separate documents for unit testing and integration testing will be developed as the project progresses.

13 Patentability Considerations

None

14 Architecture Baseline Requirements

N/A

15 Accessibility Requirements

The specific Accessibility Design Requirements (ADRs) for this program were captured in ADR checklists in the System Function Spec (SFS). The checklists contain both the accessibility requirements that will be met by this program along with accessibility requirements that were met by an earlier release of this product. This program is responsible for meeting both the new requirements and assuring that the previously met requirements continue to be met.

The ADR checklists in the SFS contain the ADR identifiers (ADR IDs) that should be listed/cited here for ease of reference during development to obtain more detailed guidance from the ADR knowledgebase on accessibility design, implementation, and testing. This information can be obtained by following the hyperlink inserted in the ADR ID. The ADR IDs that are labeled as "MUST" are required to comply with the accessibility laws. If a program drops a "MUST" requirement during development, seek guidance from Corporate Compliance Accessibility Group to determine the legal ramifications.

For more information on design for the accessibility requirements, refer to the detailed information for an ADR ID located in the ADR knowledgebase at http://wwwin.cisco.com/accessibility/requirements/. For more information on accessibility design see http://wwwin.cisco.com/accessibility/design_testing/. For more information on adding ADR IDs into the SFS, see the SFS template at EDCS-189226. Training on designing accessible software and other accessibility courses can be found at http://wwwin.cisco.com/accessibility/training/.

The list of ADRs that this Software Functional Specification addresses is the following:

<List the applicable ADR IDs from the program's System Functional Specification.>

16 Product Evolution Program, PEP

N/A

17 Requirements Traceability Considerations

<<TL 9000: ADDITIONAL REQUIREMENT FOR TL 9000 COMPLIANCE>>

Requirements traceability is required for TL 9000 registration. Traceability may be performed manually, or with assistance of automated software tools.

If the requirements in the SFS are not of sufficient detail to support traceability to test cases, you may need to trace the product requirements to the more detailed functional requirements specified in this SW functional specification. In this situation, provide unique identifiers for all functional requirements and trace product requirements to functional requirements consistent with your organization's strategy for requirements traceability. If the project has a separate traceability matrix provide a link to that document.

Refer to Common Requirements Traceability Process Handbook EDCS-400506 for a thorough description of traceability and some examples of a manual implementation

<body>

18 References

1. GSR VPLS Phase 2 Product Requirements Document, EDCS-476923
2. ITD VPLS Product Requirements Document, EDCS-461551
3. VPLS Software Functional Specification, EDCS-261721
4. GSR Engine5 VPLS Software Design Specification, EDCS-405965

19 Glossary

The following list describes acronyms and definitions for terms used throughout this document:

AC Access Circuit

CE	Customer Edge Router/Switch
DA	Destination MAC Address
H-VPLS	Heirarchical VPLS
GSR	Gigabit Switch Router
LDP	Label Distribution Protocol
MPLS	Multi Protocol Label Switching
MTU	Maximum Transfer Unit
P	Provider Router/Switch
PE	Provider Edge Router/Switch
PW	Psuedo Wire
QOS	Quality of Service
VC	Virtual Circuit
VLAN	Virtual Local Area Network
VPLS	Virtual Private LAN Services
VFI	Virtual Forwarding Instance

20 Attachments

20.1 Review Action Items

Action items are tracked using PRT # 23066.

End of Document